(12) UK Patent Application (19) GB (11) 2 271 002 (13) A

(43) Date of A Publication 30.03.1994

(21) Application No 9220379.3

(22) Date of Filing 26.09.1992

(51) INT CL[5]
G06F 15/21 13/00

(52) UK CL (Edition M)
G4A AFGX

(56) Documents Cited
None

(58) Field of Search
UK CL (Edition L) G4A AFGX AUXF
INT CL[5] G06F 7/02 13/00 15/22 15/24 15/26
ONLINE DATABASES : WPI

(71) Applicant(s)
Digital Equipment International Limited

(Incorporated in Switzerland)

1 Grand Places, 1700 Fribourg, Switzerland

(72) Inventor(s)
Charles Michael Fox

(74) Agent and/or Address for Service
Eric Potter Clarkson
St Mary's Court, St Mary's Gate, NOTTINGHAM, NG1 1LE, United Kingdom

(54) Data processing system with message preprocessing

(57) EDI (Electronic Data Interchange) systems deal with messages relating to business transactions. Message-based (batch) EDI deals with individual messages. Transaction-based EDI (I-EDI, interactive EDI) deals with transactions as wholes, messages having headers identifying their transactions. Transaction types are defined by scenarios, and individual transactions by instances of the transaction scenarios. A general-purpose transaction-based EDI system requires preprocessing means (50) for passing message-based system messages through to a message-based EDI subsystem (59). In the present system, the preprocessing means attempt (56, 57), for non-transaction messages (ie messages without transaction headers), to match them with the scenarios and instances, to thereby identify the transactions to which those messages relate.

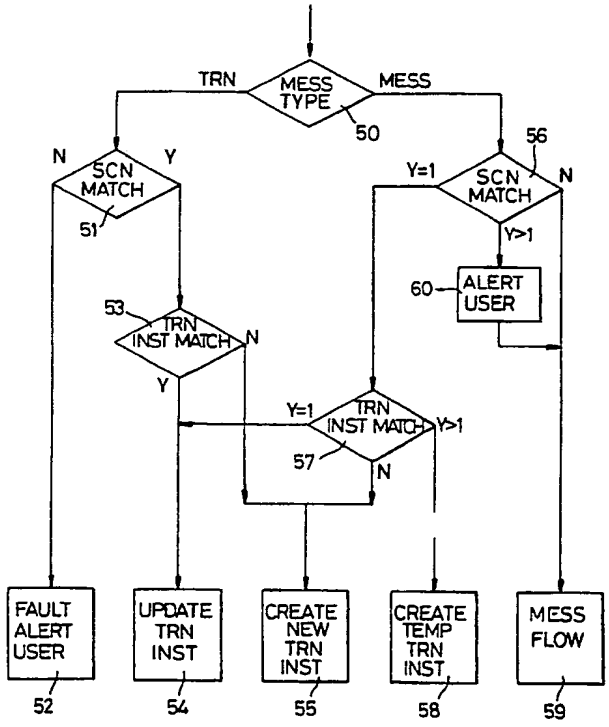

Fig. 3

At least one drawing originally filed was informal and the print reproduced here is taken from a later filed formal copy.

GB 2 271 002 A

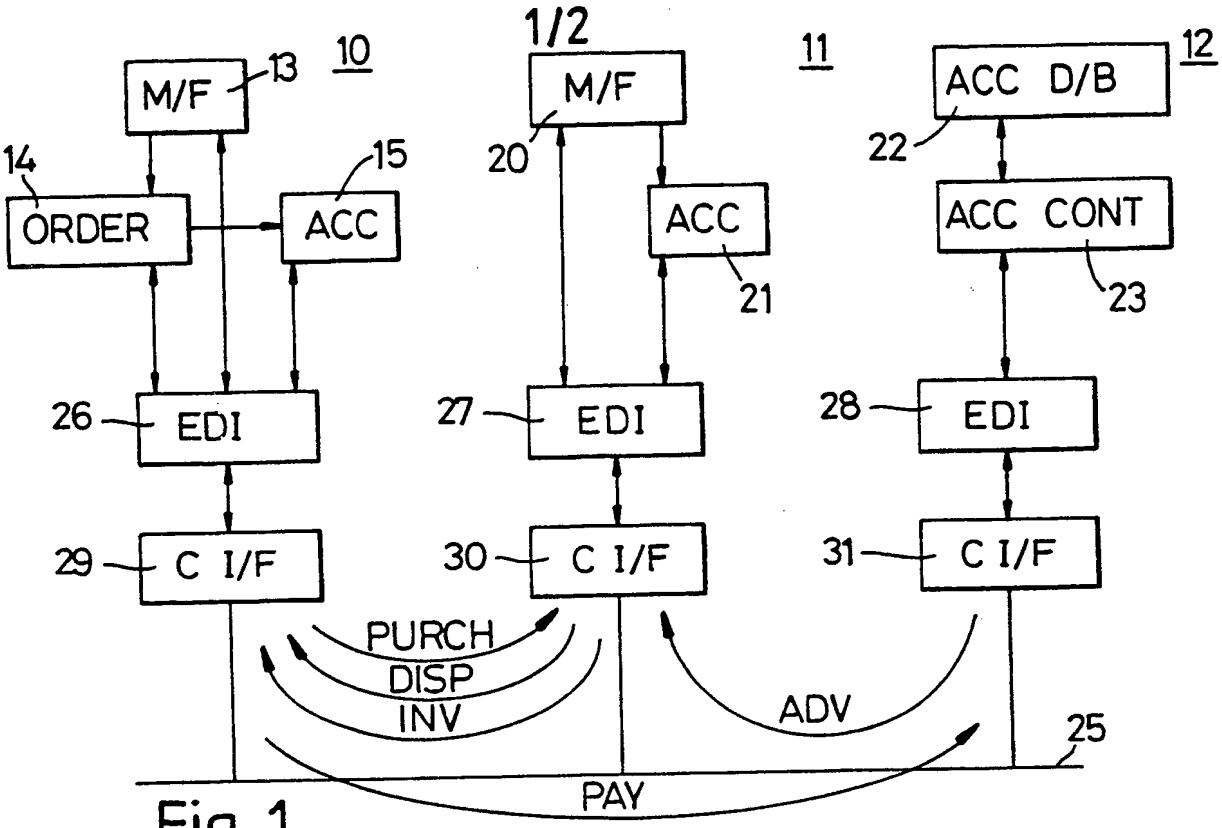

Fig. 1

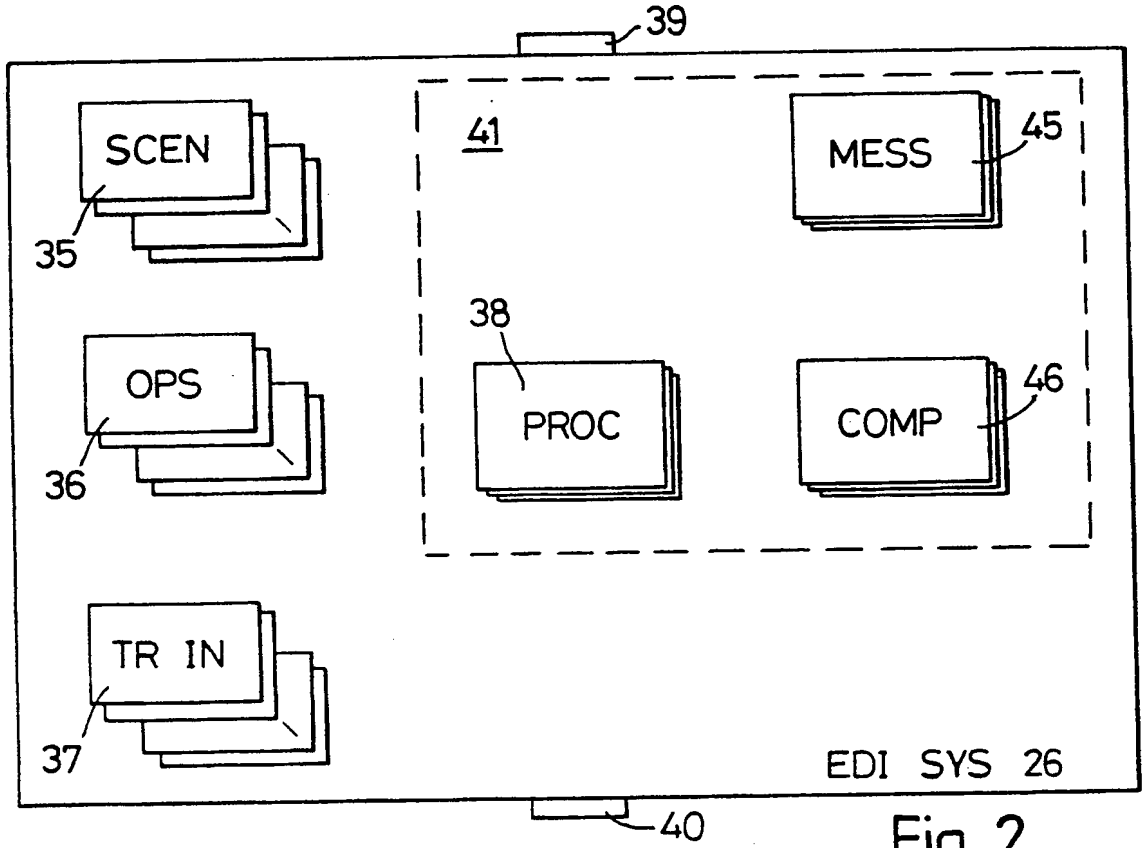

Fig. 2

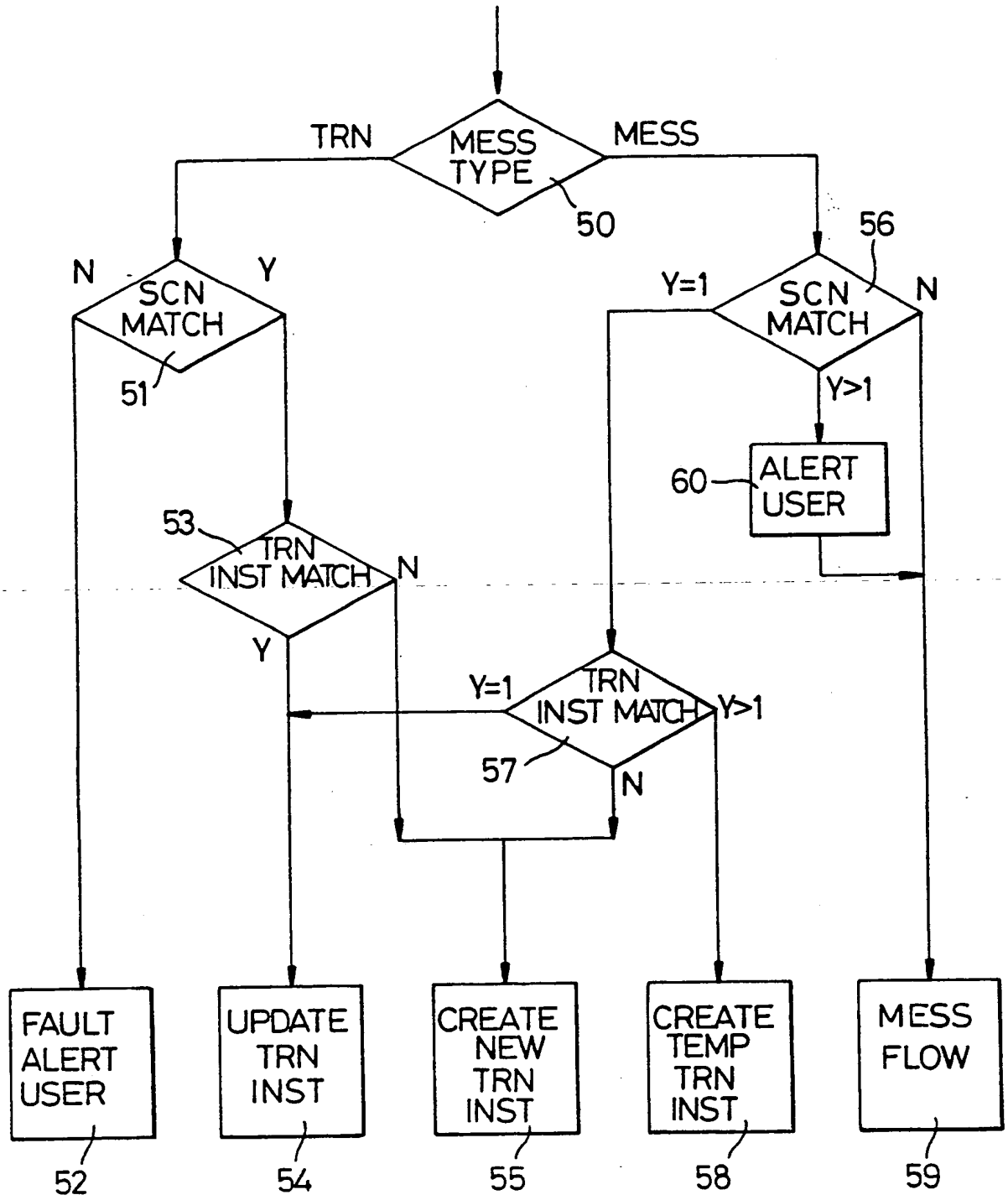

Fig. 3

2271002

Data Processing System

The present invention relates to data processing systems for monitoring and controlling business procedures, and more specifically business transactions.

General background

A typical business transaction involves several companies or, more generally, trading entities, and the interchange of a variety of documents between these entities. A reasonably simple example is a manufacturer who uses various components, including widgets, obtained from an external supplier. The manufacturer has some internal system which generates an order for a batch of widgets, and the order is duly printed out and sent off to the supplier. The supplier receives the order, and sends back an acknowledgement. In due course, the supplier makes up the order and sends it off (as a physical consignment, of course, not a document), and sends a delivery advice note at the same time. The supplier also then generates and sends an invoice. When the manufacturer has received the widgets and the invoice, he sends a payment order to his bank to debit his account and credit the supplier's account (assuming that they have the same bank); the bank then sends an advice letter to the supplier to say that the invoice has been paid.

In practice, the various trading entities will, if they are of significant

size, have formal internal systems for processing the various documents. For example, the manufacturer will have a production department which has a manufacturing control system which is fed with product orders and forecasts and generates, from that information, production targets and then component orders for the components required for those production targets. There will be a purchasing system which generates orders to suppliers for the required components, using information on potential suppliers and incorporating suitable authorization procedures. The goods received department will have procedures for reconciling goods received with advice notes. The accounts payable department will have procedures for paying invoices provided that receipt of the goods has been advised by the goods received department.

These various internal systems will often be automated to some extent; we will term such automated systems "applications" (or application systems). Thus the production department may have an application system which automatically matches information from the goods received department with original purchase orders; the accounts payable department may have an application system which automatically matches invoices with information from the goods received department; and so on. Many of the documents generated by the internal systems will be generated by such automated application systems. Some manual input may be required (eg cheque signatures), but often no manual input to the documents themselves will be required (eg in the supplier, the invoice and the delivery note can be generated automatically once

the goods have been dispatched).

The facts that some of the documents are generated automatically and that some of the documents received undergo processing that is at least partially automated have allowed a further stage of automation, known as Electronic Document Interchange (EDI). In this, the application systems of the trading entities are coupled to communication systems through respective EDI interfaces, so that documents can be sent and received automatically (ie electronically). The communications system may be a public utility service such as a telephone network, or some form of wide area network or packet switching network (which may be partially or wholly public).

The EDI service forms an interface or gateway between the applications and the communications system. For outgoing documents (or "messages"), the EDI service forwards the document from the application to the communications system; for incoming messages, the EDI service forwards the message from the communications system to the appropriate application. (In practice, there will usually be a communications system interface between the EDI service and the communications system, to convert the messages between the EDI service format and the particular characteristics of the various communications systems used.) In addition, the EDI service adds or deletes suitable headers, and may perform format conversions between internal applications formats and standardized communications message formats.

This form of EDI is message-based, and we will use the term "message-

based" (though the term "batch system" is also used for such systems). Each incoming message is treated independently by the EDI service, which inspects the message only enough to determine which application the message is to be forwarded to. Each message has a message header which contains sufficient identification for this purpose. (If the message header is missing or lacks adequate identification, the EDI service may eg send the message to a default application system for subsequent manual processing.)

ACID transaction-processing systems

There is a type of distributed data processing system known as a transaction processing system in which a "transaction" consists of a set of components all of which must be performed for the transaction to be performed. (A very simple example is the transfer of money from one account to another, where the one account must be debited and the other account must be credited; if, say, the one account were debited and the other account were not credited, the system would end up in an inconsistent state.) The acronym ACID (Atomic, Concurrent, Isolated, Durable is often used for such systems, the "atomic" indicating that the transaction is an indivisible whole.

In such systems, there are protocols to ensure that, as a transaction proceeds, the system also maintains its pre-transaction state, so that the system as a whole (ie all components of the system) "jumps" to the next state when the transaction is completed but reverts to the pre-transaction state if the transaction

(ie any component) fails (eg because some processor in the system has crashed, or a communication link has failed).

Interactive EDI

The business transactions considered up to now typically extend over long periods (months or more) and involve two or more independent parties. Both the long periods involved and the fact that the "system" is not under the control of any single party make it difficult or impossible for these transactions to be subjected to ACID-like constraints.

There are however certain specialized business systems (taking "business" in a broad sense) which are more suited to ACID-like constraints. Examples are credit card systems and booking or reservation systems. It is desirable, and often essential, for these systems to operate in "real time", and it is also essential for them to maintain consistency. (The need for consistency in banking has already been mentioned; the need for consistency in seat reservation systems is also obvious.)

Such systems have often been designed as proprietary systems. This has some advantages when there is a single user (eg a bank) using closely controlled communication systems. For systems which have multiple users (and large communications systems which are to some extent public), such debit and credit card systems, as airline ticket reservation systems (which extend over many countries and involve many independent travel agents), and

tour operators, however, standardization offers major advantages.

An extension to EDI, termed interactive EDI (I-EDI), has been proposed for such systems. Interactive EDI is concerned with business transactions (in the board sense) as a whole. Each different type of transaction is defined as a Scenario - a formal description of all the business rules and information flows of a type of business transaction among two or more parties. The scenario is in turn defined as a set of possible Dialogue types and their organization, with each dialogue involving the exchange of messages between two parties in the scenario. Interactive EDI is thus concerned primarily with extending standard EDI systems to accommodate (as its Dialogues) the elements of ACID-like transaction processing systems.

Transaction-based EDI

It will be realized, however, that Interactive EDI is defined in a manner which allows its application or extension to business transactions generally, whether or not they involve ACID-type operations. Such systems need not involve ACID-type operations and may require some elaboration of I-EDI to maximize their utility. We will use the term "transaction-based EDI" for such EDI systems.

More specifically, for transaction-based EDI, the business documents are generated in standard form (as defined by international standards). Each document includes a transaction header which identifies the transaction and the

nature of the document.

In transaction-based EDI, the EDI service itself operates as a supervisory system co-ordinating the operations of the various applications, acting to supervise the transactions. For this, the EDI service is programmed with the permissible patterns or scenarios of transactions. As each transaction proceeds, it checks that it is proceeding according to a permissible sequence or scenario. If it is, then the service forwards the individual incoming and outgoing messages of the transaction to and from the appropriate application systems, and updates its record of the transaction appropriately. In some instances, the EDI service may itself take some action; for example, it may generate a reminder message if some stage in a transaction times out. (The "message" may have a variety of forms, eg a fax message; the term "business information object" (BIO) is sometimes used for messages in this general sense.)

It should be noted that while commercial EDI systems are defined by international standards, the term EDI is used here in a general sense.

The general problem

There is a serious problem with transaction-based EDI systems as described so far. A transaction-based EDI installation can only operate effectively if there are similar transaction-based EDI installations at all the other trading entities with which it may conduct transactions. This is likely to be achievable only in the relatively distance future, if at all. Different trading

entities are likely to install transaction-based EDI services at different times; and there will be little incentive for a trading entity to install a transaction-based EDI service until a substantial proportion of the other entities with which it trades have also installed transaction-based EDI.

To overcome this problem, it is therefore desirable for transaction-based EDI systems to be designed to include a default subsystem or mode in which they operate as a simple message-based EDI system. If such a transaction-based EDI system receives a message which lacks a transaction header, it reverts to the simple message-based EDI system operation and merely forwards the message to the appropriate application system.

The invention

We have now realized that the utility of a transaction-based EDI system can be further increased by including in it means for preprocessing non-transaction messages (ie messages without transaction headers) to attempt to identify the transactions to which those messages relate. The preprocessing means perform a kind of sieving or filtering function, extracting some of the non-transaction messages and transferring them to the transaction processing system. Of course, those which cannot be identified with a transaction pass through the filter and are treated by the default non-transaction message processing subsystem.

Accordingly the present invention provides a transaction-based EDI

system comprising means for storing scenarios defining possible transactions, instance storage means for storing instances of such transactions, and transaction message processing means for processing messages relating to such instances, and further including preprocessing means for determining whether or not incoming messages are transaction messages or non-transaction messages, characterized in that the preprocessing means include non-transaction message processing means for comparing incoming non-transaction messages with the stored scenarios and instances to attempt to identify the scenarios and instances thereof to which such messages relate.

Specific embodiment

A transaction-based EDI system embodying the invention will now be described, by way of example, with reference to the drawings, in which:

Fig. 1 is a general block diagram of a group of companies coupled together by EDI systems;

Fig. 2 is a more detailed block diagram of an EDI system; and

Fig. 3 is a simplified flow diagram of the EDI system of Fig. 2.

General company interactions

Fig. 1 shows a manufacturing company 10, a supplier company 11, and a bank company 12. Each of these companies has various application systems.

Thus the manufacturer 10 has a manufacturing control application system 13, an ordering system 14, and an accounts system 15; the supplier 11 has a manufacturing system 20 and an accounts system 21; and the bank 12 has an accounts database 22 and an accounts control system 23. Various ones of these application systems may be interconnected inside each company; for example, the manufacturing control system 13 is coupled to the ordering system 14 which is in turn coupled to the accounts control system in company 10, and the accounts control system is coupled to the accounts database in company 12.

A communication system 25 (shown for convenience as a single channel, though in practice it may include LANs, WANs, and/or public telephone and other systems) connects all three companies together. Each of the companies 10-12 has certain of its application systems coupled to a respective EDI system 26-28 as shown, and the EDI systems are coupled to the communications system 25 via respective communications interfaces 29-31.

With this arrangement, a typical (simplified) transaction may involve the manufacturing application 13 in the manufacturer 10 determining that a fresh supply of widgets is required, and passing this information to the purchasing application 14, which generates a purchase order message PURCH and sends it to the supplier 11. In the supplier 11, this order is received in the manufacturing application 20. In due course, the widgets are sent out by the supplier 11 and an accompanying dispatch message DISP is generated by the

manufacturing application 20 and sent to the ordering application 14 in the manufacturer 10.

In this example, the widgets are of course sent as a physical consignment, not shown in Fig. 1. It is however possible for a transaction to consists solely of messages, if for example the subject of the transaction is abstract information such as a piece of software or a market research compilation from a manufacturing company directory or database.

Also, the accounts application 21 in the supplier 11 generates an invoice message INV and sends this to the manufacturer 10, where it is routed to the accounts application 15. This accounts application generates a payment order message PAY and sends it to the bank 12. This message is received in the accounts control application 22 in the bank, which performs the required transfer of funds from the manufacturer's account to the supplier's account in the accounts database application 23, and generates an advice message ADV which is sent to the supplier 11, where it is routed to the accounts application 21.

Although EDI systems are intended primarily for passing messages from one company to another, they can also be used for communication between different applications in a single company. In practice, the applications will normally have a number of direct linkages for passing information between them without involving the EDI system. However, the EDI system provides an alternative method of passing information, and in some circumstances it may

be more convenient to use the EDI system for that purpose than to add further direct links between the applications. In particular, the EDI system may provide a convenient way for different applications to learn of the state of the transaction, and hence of other applications.

In practice, the transaction is likely to be considerably more complicated, with more companies involved (for example, the manufacturer and the supplier will probably have different banks, a transport and delivery company may be involved, &c), and involve more messages (for example, a quotation and the approval thereof). Also, there will be considerably more interaction between the various applications in each company than have been described here (for example, reconciliations between different applications).

The description has been sufficiently general so far to apply equally to message-based and transaction-based EDI.

Transaction-based EDI system

For transaction-based EDI, each EDI system incorporates transaction descriptions which define the permissible patterns of transaction. (For convenience, we will consider only the EDI system 26 in the manufacturer 10 from now on.) We shall use the term "scenarios" for such descriptions, but it should be noted that although the present scenarios are broadly similar to the scenarios of I-EDI described above, in general they are not defined in terms of Dialogues or ACID-like operations.

Each scenario defines a permissible sequence or pattern of events. The events are primarily the reception of incoming messages, from either the applications associated with the EDI or from other companies, and the transmission of messages in response thereto. Thus in the transaction discussed above, the messages are the PURCH, DISP, INV, and PAY messages together with the associated messages passing between the EDI system 26 and the application systems 13-15. (It is also possible for the EDI system itself to generate events; for example, it may contain timers for generating messages if a particular message is not received within a predetermined time.)

The EDI system also includes means for maintaining, for each transaction, a record of the progress of that transaction. Each actual transaction is a distinct instance of a transaction scenario, and its record identifies, among other things, the appropriate scenario and the progress of the transaction through that scenario. This is required because it is possible for two separate transactions to be in existence together and to follow the same scenario; for example, a further order for more widgets may be generated while the transaction of the first order is still in progress.

Each scenario is defined as a tree of functions or elements; there is a different scenario for each different type of transaction, the scenarios being generated with the aid of a user interface (not shown). The scenarios are stored in a scenario store 35 (Fig. 2), which consists of a plurality of units each of which stores a different scenario. The nature of a typical scenario is indicated,

in somewhat simplified form, in Table I below.

In this table, some of the elements can be lists of repeated items; for example, element 3, the steps of the transaction, is a list, and the element 5.5.4 and the dependent elements will usually be lists, as most transactions will involve several different trading partners. Some elements appear more than once in the list - eg transaction steps appear in elements 3 and 5.6; in such cases, the items in the second element will identify the relevant items from the first element, which constitutes a complete listing.

The nature of the elements and items is generally self-evident from their names and the present general description. However, it may be noted that element 3 is the main steps of the transaction, and the control sequences (element 4), which broadly represent the various business functions involved, are sequences of conditions and actions. Element 5.5.4 and the dependent elements identify the trading partners in some detail. The structure of the scenario allows not merely a trading partner (company) to be specified, but also a particular business function within that company and a particular application within that business function; this allows transactions with say the various divisions of the various operating companies of a single holding company to be distinguished. Element 5.6.2 identifies a class of Business Information Object (BIO) for each step; typical classes of BIO are EDIFACT invoices, X12 purchase orders, and CAD/CAM drawings.

Table I: Business Scenario

1 Scenario ID
2 Human contact
3 Sequence of transaction steps
4 Control sequences
- 4.1 Set of conditions
- 4.2 Set of actions

5 Sequence of Information flows
- 5.1 Information flow ID
- 5.2 Direction code
- 5.3 Control sequence
- 5.4 Application
- 5.5 Trading partners
  - 5.5.1 Trading partner ID
  - 5.5.2 Human contact
  - 5.5.3 Communication services
  - 5.5.4 Business functions
    - 5.5.4.1-6 (details of business functions)
      - 5.5.4.5.1-5 (business application functions)
- 5.6 Transaction steps
  - 5.6.1 Step ID
  - 5.6.2 Associated class of BIO
- 5.7 Communication services
- 5.8 Communication links
  - 5.8.1 Communication link ID
  - 5.8.2 Human contact
  - 5.8.3 Quality of service
  - 5.8.4 Class of communication link
  - 5.8.5 Submission port address
  - 5.8.6 Delivery port address
  - 5.8.7 Elements of service
  - 5.8.8 Extensions
- 5.9 Extensions

6 Extensions

The control sequences (conditions and actions) can be regarded as part of the scenarios, but it is convenient to store them in a separate operations store 36, which consists of a plurality of units each of which stores a different set of conditions and actions. Elements 4 and 5.3 in each scenario point to the corresponding entries in store 36.

The conditions may relate to messages as a whole (eg whether a particular message has been received) or the contents of the messages (eg the contents of particular fields of the messages). The actions may similarly relate to messages as a whole (eg whether or not to send a message) or to the contents of the messages (eg changing the format of a field, checking a field to see whether its contents are within a predetermined range, checking whether acknowledgements are to be sent (or have been received), &c).

Turning now to the instances of the transactions, there is a transaction instance store 37 which consists of a plurality of units each of which stores a different instance table. There is a different instance table for each of the current transactions, ie transactions which are actually proceeding. The structure of each instance table is defined by the scenario of the transaction involved. Since there may be several current transactions for a given scenario, there may be several different instance tables defined by the same scenario, each indicating the current state of a different one of those transactions.

Each instance table is defined as a table of transaction steps, with each

transaction step consisting of a step number, a step state, a set of step conditions, and a set of actions. The table also has an identifier, which may loosely be termed a transaction identifier but is more precisely a transaction instance identifier. For each step, the main possible step states are pending, active, or completed (though other states such as blocked and paused are also possible).

The EDI system also includes a processing unit 38, which is coupled to the scenario store 35, the operations store 36, the transaction instance store 37, a set of internal ports 39 to the applications 13-15, and an external port 40 to the communications system 29. This processor operates to maintain the instance tables and carry out the appropriate actions from the operations store 36. In effect, each scenario is an abstract state machine, and the processor, the scenarios, and the transaction instance tables together implement, for each transaction instance, this state machine and its advance through its various states.

Each message in a transaction-based EDI system has an EDI envelope or header. The EDI system generates the envelopes for messages which it sends out, and examines and strips off the headers of messages which it receives. (Some messages pass through the EDI system from or to the applications systems; others are generated or terminate in the EDI system itself.) The header includes identification information to identify the transaction involved. This may take various forms; the preferred form of transaction

identifier comprises a scenario identifier, a step (of the scenario) identifier, and an instance identifier.

The generation of the transaction identifier for outgoing messages is straightforward. For incoming messages, the processor analyzes its header to identify the transaction identifier, and from that, to obtain the scenario, step, and instance identifiers. It then uses that information to refer to the appropriate scenario table in store 35 and determine what operations need to be performed (looking them up in table 36). It then carries out those actions and those determined by the active state in the transaction instance table, and updates the transaction instance table.

This system works for messages which have transaction instance identifiers; that is, for messages which are received from companies which also have transaction based EDI systems. However, as discussed above, for many transactions at least some of the other companies involved may have only message-based EDI systems. Messages received from those companies will lack transaction instance identifiers.

Message identification

In a simple transaction-based EDI system, the EDI system would either pass such messages through directly to the application systems (without updating involving the transaction control means described with reference to Fig. 2) or would pass them to an exception or invalid message routing for

subsequent processing by the user. In the present system, however, the EDI system includes means 41 for attempting to identify the transactions to which such simple messages (ie messages without standard transaction headers, ie explicit instance identifiers) relate.

For message identification, the message is stored in a message store 45, and is processed by the processor 38 operating in conjunction with a comparator unit 46. A simplified version of the operation of the system - that is the general processing of messages, whether or not they are transaction-based - is summarized by the flow diagram of Fig. 3.

In the first operation, block 50, the processor inspects the header of the message to determine whether it is a simple message header (MESS) or a transaction header (TRN).

If it is a transaction header, block 51 follows, in which the processor extracts the scenario identifier from the transaction identifier and tries to match it against the identifiers of the scenarios in the scenario store 35, using the comparator 46. If the match fails (N), then the system signals a fault and alerts the user (block 52). If the match succeeds (Y), then block 53 follows, in which the processor extracts the instance identifier from the transaction identifier and tries to match it against the identifiers of the instances in the instance store 37, using the comparator 46. If a match is achieved (Y), the system proceeds to block 54, in which the transaction is advanced - the transaction is updated, the appropriate actions taken, &c. If a match is not achieved (N), then the system

proceeds to block 55, in which a new transaction instance is created and the appropriate actions taken.

If the message does not have a transaction header, then it is a simple message (ie from a message-based EDI system). In a simple transaction-based EDI system, the flow diagram would be as described so far, but the MESS output from block 50 would lead straight to a message-based EDI system, block 59. In the present system, however, the system proceeds to block 56, in which the processor uses the comparator 46 to try to match the message with a scenario.

If the comparison of block 56 is successful with a single scenario (Y=1), the system the proceeds to block 57, in which it tries to match the message with a transaction instance. If the match succeeds against a single transaction instance (Y=1), then the system has successfully identified a transaction instance for the message, and proceeds to block 54 as before. If the match fails (N), then the system proceeds to block 55, also as before. If the match succeeds against multiple transaction instances (more than one instance) (Y>1), then the system proceeds to a block 58, in which a temporary transaction instance is created. The possible instances are recorded with the temporary instance, and the ambiguity will eventually be resolved as the various transactions proceed further.

If the comparison of block 56 is unsuccessful (N), the system is unable to identify the message with a transaction, and passes the message on to the

message-based EDI system (block 59). The final possibility for block 56 is more than 1 successful comparison ($Y > 1$); in this event, the system alerts the user (block 60) and then proceeds to block 59 as before.

It will be realized that although most of the blocks of the flow shown in Fig. 3 involve comparisons, the particular comparisons differ from block to block, and the comparisons involved in processing message-based messages are in general more elaborate than those involved in the transaction-based messages. For the message-based message processing, the comparator is preferably a pattern matching unit which attempts to find a sufficient match between the message and one of the scenarios in store 35 or transaction instance tables in store 37.

For example, for scenario matching the comparator may attempt to match the trading company identified in the message in the store 45 with the trading companies identified in the various scenarios stored in store 35. This matching is of course be carried to the appropriate level (trading partner (company), particular business function within that company, or particular application within that business function) as appropriate.

If a trading partner is identified, then this may determine a scenario. The comparator 46 may then attempt to match the message to a particular message type in the scenario, by matching the characteristics of the message with those of the various message types in the scenario.

If a matching message type is identified, then the comparator may

attempt to match the message with any of the transaction instance tables in store 37 for the possible scenario and message type. If this is successful, then the transaction instance for the message has been successfully identified, and the message is then processed in the manner described above.

If desired, further matching tests may be performed, dependent on the particular message type. For example, if the message is identified as being an invoice message INV described above (Fig. 1), this will have an internal order identifier generated by the supplier 11 which will be identical to the same order number in the preceding dispatch message DISP from the same company. A check can therefore be made for this match.

A message in general will consist of some kind of envelope or header and some contents or data. Thus a message from a message-based EDI system will have a message-based EDI header, and a message which passes through a communication system will have a communications header, and these headers may well contain identifiable information about eg the business origin of the message. The present matching system can usefully endeavour to match on the context information of the message, ie information from the header or headers, as well as the information in the actual data content of the message.

It will be realized that for successful matching, the scenarios and transaction instances must contain an adequate amount of information about the transactions - more than is required for a transaction-based EDI system alone. The scenarios description format shown in Table I therefore contains a

considerable amount of detail about the transactions - a good deal more than is contemplated in current proposals for transaction-based EDI systems.

Obviously, the details of this matching can be varied. For example, the $Y > 1$ output from block 56 can be followed by block 57, with the transaction instances for each possible scenario being tested in turn for possible matches. The outputs from block 57 can remain unchanged, or the $Y > 1$ output may then be taken to block 59 or 60.

Claims

1 A transaction-based EDI system comprising means (35) for storing scenarios defining possible transactions, instance storage means (37) for storing instances of such transactions, and transaction message processing means (38) for processing messages relating to such instances, and further including preprocessing means (50) for determining whether or not incoming messages are transaction messages or non-transaction messages, characterized in that the preprocessing means include non-transaction message processing means (46; 56, 57) for comparing incoming non-transaction messages with the stored scenarios and instances to attempt to identify the scenarios and instances thereof to which such messages relate.

2 A transaction-based EDI system according to claim 1, characterized in that each scenario consists of a tree of functions (Table I) and is stored in a scenario store (35), which consists of a plurality of units each of which stores a different scenario.

3 A transaction-based EDI system according to claim 2, characterized in that the scenarios include control sequences (Table I, items 4 and 5.3), consisting of sequences of conditions and actions.

4 A transaction-based EDI system according to claim 3, characterized in that the control sequences (conditions and actions) are stored in a separate operations store (36), which consists of a plurality of units each of which stores a different set of conditions and actions.

5 A transaction-based EDI system according to any previous claim, characterized in that the scenarios identify trading partners (5.5, Table I) as a trading partner (company) to be specified, a particular business function within that company, and a particular application within that business function.

6 A transaction-based EDI system according to any previous claim, characterized in that the instance storage means (37) comprises a plurality of units each of which stores a different instance table including data indicating the current state of the corresponding transaction instance.

7 A transaction-based EDI system according to any previous claim, characterized in that the non-transaction message processing means (46; 56, 57) include means (56, 57) for distinguishing between match failure (N), a unique match ($Y=1$), and a multiple match ($Y>1$) of scenarios and/or instances.

8 A transaction-based EDI system according to claim 7, characterized in that the non-transaction message processing means (46; 56, 57) include means

(58) for creating temporary transaction instances in response to multiple matches.

9 A transaction-based EDI system according to any previous claim, characterized in that the non-transaction message processing means (46; 56, 57) include a pattern matching unit.

Patents Act 1977 Examiner's report to the C roller under Section 17 (The Search report)	Application number 9220379.3
Relevant Technical Fields (i) UK Cl (Ed.L) G4A AFGX, AUXF (ii) Int Cl (Ed.5) GO6F 7/02, 13/00, 15/22, 15/24, 15/26	Search Examiner PAUL NICHOLLS Date of completion of Search 27 OCTOBER 1993
Databases (see below) (i) UK Patent Office collections of GB, EP, WO and US patent specifications. (ii) ONLINE DATABASE:WPI	Documents considered relevant following a search in respect of Claims :- 1-9

Categories of documents

X: Document indicating lack of novelty or of inventive step.

Y: Document indicating lack of inventive step if combined with one or more other documents of the same category.

A: Document indicating technological background and/or state of the art.

P: Document published on or after the declared priority date but before the filing date of the present application.

E: Patent document published on or after, but with priority date earlier than, the filing date of the present application.

&: Member of the same patent family; corresponding document.

Category	Identity of document and relevant passages	Relevant to claim(s)
	NONE	

Databases: The UK Patent Office database comprises classified collections of GB, EP, WO and US patent specifications as outlined periodically in the Official Journal (Patents). The on-line databases considered for search are also listed periodically in the Official Journal (Patents).

BNSDOCID: <GB___2271002A__I_>